Open Problems in Computer Virology

Eric Filiol, Marko Helenius, and Stefano Zanero

[1] Ecole Supérieure et d'application des Transmissions
Laboratoire de virologie et de cryptologie
B.P. 18, 35998 Rennes, France
efiliol@esat.terre.defense.gouv.fr
[2] University of Tampere
Department of Computer and Information Sciences
Kanslerinrinne 1
FIN-33014 University of Tampere - Finland
cshema@cs.uta.fi
[3] Politecnico di Milano
Dip. Elettronica e Informazione
Via Ponzio, 34/5
I-20133 Milano - Italy
zanero@elet.polimi.it

Abstract. In this article, we briefly review some of the most important open problems in computer virology, in three different areas: theoretical computer virology, virus propagation modeling and antiviral techniques. For each area, we briefly describe the open problems, we review the state of the art, and propose promising research directions.

1 Introduction

Research in computer virology is still somehow controversial. A widely spread misconception believes that researching on computer virus propagation is neither interesting, nor productive: it is potentially dangerous, since it can lead to the development of more devastating techniques of viral infection, and in any case it is just a waste of time, because the job of fighting computer viruses is limited to the "catch, analyze, deploy signatures" cycle typical of the anti-virus industry.

This widespread belief explains why there are just a few research teams in universities and research organizations worldwide that deal with computer virology. An insufficient dissemination and knowledge of the few remarkable theoretical results that have been obtained until now in this field partly accounts for this belief. Upon closer examination, these results demonstrate that, on the contrary, a deep research in computer virology is absolutely urgent and essential.

In order to maintain a relatively efficient protection of our systems, in order to try and anticipate computer viral hazards before they actually

materialize in the hands of attackers and malware writers, we need to understand in depth the threat we are facing, and how it is evolving. We cannot rely on a "wait and see" approach, but we must anticipate technological evolutions.

Unfortunately, many open problems still exist as far as computer virology is concerned, in both theoretical and technical aspects. Many other problems will doubtlessly emerge in the future, due to the ingenuity of malware writers. In the meantime, computer systems become more and more complex, more and more sensitive, making old virus protection and defense models progressively inadequate.

The purpose of this paper is to present what we believe to be the most interesting open problems in computer virology. We selected the problems whose resolution, or in-depth study, is likely to generate a valuable contribution to the level of the field, and also to improve the quality of detection and protection applications. Also, we tried to focus on theoretical problems in computer virology, which can motivate scholars in theoretical computer science or in mathematics to research in this field. Computer virology is not just an endless hunt between virus coders and antivirus labs, but offers a lot of theoretically deep problems to fathom.

We focused on four major aspects, which correspond to the general organization of the paper. Section 2 deals with open problems in theoretical computer virology. Founding fathers of this field, like Fred Cohen or Leonard Adleman, have produced essential theoretical results, thus giving birth to computer virology as we know it. But their seminal works have opened up other interesting problems that are still to be solved. Another aspect comes from the fact that the theoretical models they proposed tend to become unsuitable to describe some new viral risks. Many complexity issues still require researchers' attention. Classification aspects are worth considering in order to help to clearly identify the true and complete nature of what the computer viral hazard is and how it may evolve in the future.

Section 3 considers open problems in virus propagation modeling techniques. We review the mainstream literature works on this topic, and show why new modeling techniques are needed to capture new trends in the propagation of common viruses, mass-mailers, random scanning worms of new conception, and we also briefly deal with various basic issues surrounding propagation modeling.

Section 4 deals with proposed countermeasures: how can they be validated before being implemented? Which new defensive techniques do we

need to counter new developments we foresee in the next generations of aggressive malware?

Section 5 presents instead some practical and technical research areas that could benefit from a theoretically sound scientifical approach which is currently lacking.

Finally, in Section 6 we draw conclusive remarks on this review, and outline the most interesting issues for future research in the area.

2 Open Problems in Theoretical Computer Virology

2.1 Theoretical Definitions of Viruses

Let us recall the different theoretical definitions for computer viruses that have been proposed in previous research. It will help the reader to better understand what follows.

– Cohen's definition considers Turing machines [1]. The basic notion is that of viral set.

Definition 1 *(Viral set)*
For all Turing machines M and all non-empty sets of Turing programs V, the pair (M, V) is a viral set*, if and only if, for each virus $v \in V$, for all histories of the machine M, we have:*

- *For all time instants $t \in \mathbb{N}$ and cells j of M if*
  1. *the tape head is in front of cell j at time instant t and*
  2. *M is in its initial state at time instant t and*
  3. *the tape cells starting at index j holds the virus v,*

  then, there exists a virus $v' \in V$, at time instant $t' > t$ and at index j' such that
  1. *index j' is far enough from v position (start location j),*
  2. *the tape cells starting at index j' hold the virus v' and*
  3. *at some time instant t'' such that $t < t'' < t'$, v' is written by M.*

In an abridged way, we can write that V is a viral set with respect to M, if and only if,

$$[(M, V) \in \mathcal{V}]$$

and that v is a virus *with respect to M, if and only if,*

$$[v \in V] \text{ such that } [(M, V) \in \mathcal{V}].$$

In this context, a "simple" virus can be described by a singleton viral set.

– Adleman's definition as well as Zuo and Zhou's one relies on recursive functions [2, 3] (we consider here the formalism adopted in [4] for the purpose of homogeneity with the next definition).

Definition 2 (Adleman's viruses) *A total computable function $\mathcal{A}$ is said to be an A-viral function (virus in the sense of Adleman) if for each system environment (r, d), one of the three following properties holds:*

Injure

$$\forall p, b \in \mathcal{D} \quad \varphi_{\mathcal{A}(p)}(r, d) = \varphi_{\mathcal{A}(q)}(r, d). \tag{1}$$

This item corresponds to the execution of some viral functions independently from the infected program.

Infect

$$\forall p \in \mathcal{D} \quad \varphi_{\mathcal{A}(p)}(r, d) = \langle \varepsilon_{\mathcal{A}}(r'_1), \ldots, \varepsilon_{\mathcal{A}}(r'_n), d' \rangle \tag{2}$$

where $\varphi_p(r, d) = \langle r', d' \rangle$ and $\varepsilon_{\mathcal{A}}$ is a computable selection function defined by

$$\varepsilon_{\mathcal{A}}(p) = \begin{cases} p \text{ or} \\ \mathcal{A}(p) \end{cases}$$

The second item corresponds to the case of infection (any program is potentially rewritten according to $\mathcal{A}$; data are left unchanged).

Imitate

$$\forall p \in \mathcal{D} \quad \varphi_{\mathcal{A}(p)}(r, d) = \varphi_p(r, d). \tag{3}$$

The last item corresponds to mimic the original program (stealth purpose).

where $\mathcal{D}$ denotes the computation domain. The reader will note that this definition is not constructive, as opposed to the next one.

– Bonfante, Kaczmarek and Marion [5, 4] describe viruses as fixed points of a semi-computable function. They first consider the following definition:

Definition 3 *Assume that $\mathcal{B}$ is a computable function. A virus with respect to $\mathcal{B}$ is a program v such that for each p and x in the computation domain $\mathcal{D}$,*

$$\varphi_v(p, x) = \varphi_{\mathcal{B}(v,p)}(x).$$

The function $\mathcal{B}$ is called the propagation function of the virus v.

Then, the authors proved the following result:

Theorem 1 *Given a semi-computable function f, there is a virus v such that for any p and x in $\mathcal{D}$, we have*

$$\varphi_v(p, x) = f(v, p, x).$$

Recursion Theorem provides a fixed point v of the semi-computable function f. This fixed point v is a virus with respect to a propagation function $\mathcal{B}(v, p)$.
One of the most interesting characteristics of this approach is that such definitions and results are of constructive nature (in particular the reader will consider [4, Section 4.6, Theorem 4]).

2.2 Complexity Theoretic Problems

Studying complexity aspects of viral sets is of high importance since it quantifies the intractability of detection. Very few papers have been focused on the intractability of detection even if some major results have been established. Fred Cohen [1] proved that the general problem of viral detection was undecidable. This result refers to computability as presented by Rogers [6]. Most of the results on viruses concern undecidability and the hierarchies on the top of the Halting problem.

Later on, his Ph.D. tutor L. Adleman [2] gave complexity results on some particular instances of the general detection problem:

- The set $V = \{i | \Phi_i \text{ is a virus}\}$ is Π_2-complete [2, p. 363].
- The infected set of a virus v defined as $I_v = \{i \in \mathbb{N} | (\exists j \in \mathbb{N}[i = v(j)]\}$ is Σ_1-complete [2, p. 371].

D. Spinellis proved in 2003 that detection of bounded-length polymorphic viruses is a NP-complete problem [7]. When considering polymorphic viruses of (possibly) unbounded length, how does the detection complexity change? Such a question may appear only of theoretical interest,but in fact k-ary viruses (see Section 2.3) can simulate this behavior. More recently, a few additional results have been published:

- In 2004, Z. Zuo and M. Zhou [3] have exhibited viral sets that are Σ_1-complete, Π_2-complete or Σ_3-complete. Moreover, they also considered other viral sets that appear to be of even higher complexity.
- In 2005, G. Bonfante, M. Kaczmarek and J.-Y. Marion [4] gave other similar results.

All these results refer to algorithmic complexity as considered in [8]. In this context, research is focused on classes of low complexity where either time or space are bounded.

Despite the fact that most of this theoretical results prove that the related detection problems are intractable, in practice it remains essential to identify classes of viral codes that effectively challenge protection policies. An interesting problem is to determine whether there exist viral sets of Π_n or Σ_n complexity (complete or not) for any given value of n. From an intuitive point of view, the answer seems to be positive. Some new examples of viral codes suggests it. To carry matters to extremes, one could in fact consider indecidability as the infinite complexity ($n \to \infty$).

The answer to the previous problem in fact appeals to another problem: is it possible to classify viral codes according to the complexity class of their viral sets? Up to now, viral classification has been established by considering mathematical tools (Turing Machine [1], recursive functions [2, 3], or fixed points of a semi-computable function [4, 9]; see Section 2.1). Classifications based on complexity, rather than on mathematical properties, could produce a better perception of the viral risk and hence new models for antiviral research. The classification according to detection complexity should help to better identify classes of viruses for which detection is of polynomial complexity. This approach was first suggested in [4, Theorem 14].

Recently Z. Zuo and M. Zhou [10] presented new results on time complexity of computer viruses (virus running time, virus detection procedure). The authors pointed out some interesting open problems related to the time complexity issue. Their main results are:

- For any type of computer viruses, there exists a computer virus v whose infecting procedure has arbitrarily large time complexity.
- For any type of computer viruses, there is a virus v such that any implementation of v can have arbitrarily large time complexity in its infection procedure.

It is a well-known result [11] that there exists a computer virus v such that its infected programs set I_v is undecidable. This can formally be expressed by the fact that I_v is a non recursively enumerable set. Thus detecting all the programs infected by v requires to find a recursive set $\mathcal{C}$ such that $I_v \subset \mathcal{C}$.

Considering the fact that existing computer viruses are almost always decidable, the authors of [10] then considered two unsolved questions:

1. If I_v is decidable, what is its time complexity?

2. If I_v is undecidable, what is the time complexity of the recursive set containing I_v?

They gave only a partial answer to the first question. They proved that for any undecidable computer virus, there is one detecting procedure of arbitrarily large complexity. As the authors noted in their article, in practice it is more desirable to consider the existence of a recursive set $\mathcal{C}$ such that $I_v \subseteq \mathcal{C}$ and whose characteristic function has a "low-time" complexity (polynomial). While this is trivial when $\mathcal{C} = \mathbb{N}$, it is still an open problem to solve under the conditions that $(\mathbb{N} - \mathcal{C})$ is infinite and $\mathcal{C}$ is as small as possible.

2.3 Viral and Antiviral Models Problems

Some recent viruses – found in the wild or studied as part of a prospective protection strategy – exhibit new structures, properties and/or behaviors. Most of the time, these viruses pose new threats that current antiviral models cannot deal with. The reason is that these new viruses develop a complex, sophisticated algorithmic that does not fit to the present viral models. A good example are the so called k-ary viruses (sometimes denoted as *combined viruses* or *viruses with "rendez-vous"*). These viruses combine their respective actions according to different modes of operation. A known example of a 2-ary virus is the combination of the *W32.Qaz* virus with the *W32.Funlove* virus.

Despite the fact that their attack scheme was not very sophisticated compared to what 2-ary viruses can theoretically do, this combination illustrates a new face of tomorrow's threats. In [9, pp. 135ff], a classification of this type of viruses has been sketched. Some particular types are exhaustively presented, from an algorithmic point of view, in [12]. However, a complete and exhaustive categorization of all types of k-ary viruses and of their modes of combined action is still missing.

The difficulty of studying these particular viruses comes from the fact that they do not comply with existing models of computer viruses. As of now, computer virus models rely on the concept of "univariate" recursive functions $f : \mathbb{N} \to \mathbb{N}$. Unfortunately, these functions do not take into account, among many other aspects, the time indexing which is an inherent characteristics of k-ary viruses due to some of their modes of operation (their respective action may occur with a different time reference or index). Multivariate vector recursive functions $f : \mathbb{N}^k \to \mathbb{N}^k$ could be considered instead, in order to capture the concept of k-ary viruses. Three questions arise:

- Is a model based on multivariate vector recursive functions the best possible one for k-ary viruses? Considering the family of functions $(f_i)_{1\leq i\leq k}$ with $f_i : \mathbb{N} \to \mathbb{N}$ could produce a more general and more efficient model, being a third-order logic model.
- Will these models help to identify previously unforeseen classes of viruses?
- What kind of corresponding antiviral models do we have to develop and what are the new complexity issues with respect to them?

The next point deals with the classification of viral models themselves and their respective relationship. Existing models (Cohen's model based on Turing machines, Adleman's model based on recursive functions, and Bonfante et al.'s model based on solutions of a fixed point equation) are all second-order logic models, and have been proven to be largely equivalent. Antiviral models that have been built from them are equivalent too, and therefore are not different in their detection capabilities.

If we consider to create new viral models, let us call them $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$, we can ask ourselves:

- Do we have a logical chain for all of them, that is to say $\mathcal{M}_1 \prec \mathcal{M}_2 \prec \ldots \mathcal{M}_n$? In this context, each new model $\mathcal{M}_{n+1}$ yields a generalization of the antiviral models that have been derived from previous viral models.
- On the contrary, do we have a lattice structure for the viral models ? In this case, there exists a finite number of pairs of models that are not comparable. In other words, for some pairs $\mathcal{M}_i, \mathcal{M}_j$, neither $\mathcal{M}_i \prec \mathcal{M}_j$ nor $\mathcal{M}_j \prec \mathcal{M}_i$. In this context, we have the same organization for corresponding antiviral models. This implies a totally different, more challenging, management of viral detection.

2.4 Classification and Identification Problems

The identification of new viral classes that may represent futuer threats is essential. This identification is quite always reactive, since it relies on code analysis. Another approach is to mathematically forecast new viral techniques or classes. As a representative example, Zuo and Zhou [3] have proven that polymorphic viruses with infinite forms exist. But until now, no such viruses have been created in the real world, excluding the trivial polymorphic viruses, e.g. the padding function [13]. It remains an open problem to determine whether this computability paradigm would produce non-trivial polymorphic viruses when considering real programs.

Does Zuo and Zhou's class of specific polymorphic viruses effectively represent a practical risk? This problem may sound very provocative (in fact it would require us to write a virus), but only the proof-by-experience can give a definitive answer.

As far as polymorphic and metamorphic viruses are concerned, the classification of the mutation process is also an open problem. Detection is mostly based on heuristic techniques and their efficiency is regularly defeated by new mutation techniques. Let us recall that detection of bounded-length polymorphic viruses is a NP-complete problem [7]. In order to improve detection of poly/metamorphic viruses a new approach has to be found. Formally, Zou and Zhuo [3, 10] have defined (following Adleman) polymorphic viruses as follows:

Definition 4 *(Polymorphic virus with two forms) The pair (v, v') of two different total recursive functions v and v' is called a polymorphic virus with two forms if for all x, (v, v') satisfies*

$$\phi_{v(x)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_x(d, p[v'(S(p))]), & \text{if } I(d,p) \\ \phi_x(d,p), & \text{otherwise} \end{cases}$$

and

$$\phi_{v'(x)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_x(d, p[v(S(p))]), & \text{if } I(d,p) \\ \phi_x(d,p), & \text{otherwise} \end{cases}$$

Real-life polymorphic viruses are then described by the two authors as a n-tuple $(v_1, v_2, \ldots, v_n)$ of n different total recursive functions, under similar condition as in Definition 4. Metamorphic viruses are defined in much the same way, except that two selection functions $S(p)$ and $S'(p)$, which choose a program p to infect, are used instead of only one for polymorphic viruses. With this formalism, only a set-theoretic, computability approach is considered. In this context, this approach clearly relates to Cohen's formalism (concept of *Largest Viral Set with respect to a Turing machine*). The main drawback with the set approach comes from the fact that relationships between the evolved forms do not appear very clearly. On the contrary, polymorphism (and metamorphism) is generally and practically implemented as an algorithm that iterates over the different mutated forms. The function may be very complex (like cellular automata). In other words, polymorphism and metamorphism should be described by a functional approach rather as a viral set containing the different evolved forms of a given virus.

By considering the recursion theorem and the approach presented in [9, Chap. 1] and developed in [5, 4], we can think of a virus as a fixed point of the equation

$$\varphi_e(p, x) = f(e, p, x).$$

Then the functional description of polymorphism enables to see the i-th evolution of a virus v as the result of total recursive function f, iterated i times. In other words, we have now to consider the equation

$$\varphi_e(p, x) = f^i(e, p, x).$$

Such a modeling opens interesting and unreseolved problems, whose solution could provide a significant improvement in polymorphism detection:

- Is it possible to find some mathematical properties for the function f which could help to precisely characterize what polymorphism really is and to classify functions realizing code polymorphism (see [Remark 17]bkm)? We could imagine, as an example, some distance d between $f^{i-1}(e, p, x)$ and $f^i(e, p, x)$ which could reveal interesting invariant or probabilistically invariant properties. A first idea suggests to describe things in terms of function orbit and to focus on orbit properties.
- From a practical point of view, each evolved form of a virus can be described as a binary sequence v_i, that is to say as a codeword of length L where L is the size of each evolved form. Without loss of generality, we can consider code mutation to be size-invariant, since generalization to code size variation is straightforward. Then, the set of all mutated forms can be described as a code of length L (see [14]). Then:
  - What is the code cardinality? This relates to the number of possible evolved forms. Obviously, the code cardinality is upper-bounded by 2^L, but since any codeword of length L does not systematically represent a viable form from an execution point of view, the cardinality of the code is bound to be strictly less that 2^L.
  - What is the code minimal distance? What is the average Hamming distance between two codewords (evolved forms) of a virus?
  - How mathematical tools of coding theory could be used and applied to help in mutation process characterization and detection?
- Considering the last point of the preceding item, we could for example use tools taken from signal processing (when considering a virus as a binary sequence or an octal sequence): the discrete cross-correlation function to measure similarity (at least from a probabilistic point of

view) between to evolved forms. Knowing a given viral form, cross-correlation would probably help to find evolving features in an unknown (probably evolved) viral sequence. Autocorrelation function [9, Chap. 8, Exercises] is a noteworthy tool which can help detect some similarities inside a code and thus reveal some repetitive (dummy) code insertion for polymorphic purposes. Many discrete transforms commonly used in signal processing and coding theory should be considered as well to study and reveal mathematical properties of the iterated function describing a mutation process.

Another interesting problem deals with the impact of quantum computing [15] on computer virology. With quantum computing many research fields have made essential progress. The best example is probably cryptography where problems of intractable complexity (for traditional computers) can be solved very easily by means of a quantum computer [16, 17] The problem is two-fold:

- Considering intractable viral detection problems, what would be the impact of quantum computing? Is it possible to imagine quantum viral detection algorithms ("quantum antivirus")?
- Considering quantum computers, what would then be a quantum computer virus? Consequently, what would be the effect of such a virus in terms of detection capabilities, when processed by a quantum antivirus?

3 Open Problems in Virus Propagation Modeling Techniques

3.1 Need and requirements for propagation models

Creating reliable models of virus and worm propagation is beneficial for many reasons. First, it allows researchers to better understand the threat posed by new attack vector and new propagation techniques. For instance, the use of conceptual models of worm propagation allowed researchers to predict the behavior of future malware, and later to verify that their predictions were substantially correct [18].

In second place, using such models, researchers can develop and test new and improved models for containment and disinfection of viruses without resorting to risky "in vitro" experimentation of zoo virus release and cleanup on testbed networks [19].

Finally, if these models are combined with good load modeling techniques such as the queueing networks, we can use them to predict failures

of the global network infrastructure when exposed to worm attacks. Moreover, we can individuate and describe characteristic symptoms of worm activity, and use them as an early detection mechanism.

In order to be useful, however, such a model must exhibit some well-known characteristics: it must be accurate in its predictions and it must be as general as possible, while remaining as simple and as low-cost as possible. The importance of this work, and the shortcomings of many existing models, are described in [20].

3.2 Open questions in modeling traditional viruses

Viral code propagation vectors have evolved over the years, and propagation models also have evolved to keep pace. In the beginning of the virus era, viruses infected host programs, and the most common vector of propagation was the exchange of files via magnetic supports. The same concept, in more recent times, has been extended to macro languages embedded in office automation suites, generating the so-called "macro viruses".

The first complete application of mathematical models to computer virus propagation appeared in [21]. The basic intuitions of this work still provide the fundamental assumptions of most computer epidemiological models. Epidemiological models abstract from the individuals, and consider them units of a population. Each unit can only belong to a limited number of states (Table 1 reports a widely accepted nomenclature): usually, the name of a model explicits the chain , e.g., a model where the Susceptible population becomes Infected, and then Recovers, is called a SIR model.

Another typical simplification consists in avoiding a detailed analysis of virus transmission mechanics, translating them into a probability that an individual will infect another individual (with some parameters). In a similar way, transitions between other states of the model are described by simple probabilities. Such probabilities could be calculated directly by the details of the infection mechanism or, more likely, they can be inferred by fitting the model to actual propagation data. An excellent analysis of mathematics for infectious diseases in the biological world is available in [22].

Most epidemiological models, however, share two important shortcomings: they are *homogeneous*, i.e. an infected individual is equally likely to infect any other individual; and they are *symmetric*, which means that there is no privileged direction of transmission of the virus. The former makes these models inappropriate for illnesses that require a non-casual

M	Passive immunity
S	Susceptible state
E	Exposed to infection
I	Infective
R	Recovered

Table 1. Typical states for an epidemiological model

contact for transmission; the latter constitutes a problem, for instance, in the case of sexually-transmitted diseases.

In the case of computer viruses both problems are often present. Most individuals exchange programs and documents (by means of e-mails or diskettes) in almost closed groups, and thus an homogeneous model may not be appropriate. Furthermore, there are also "sources" of information and programs (e.g. computer dealers and software distributors) and "sinks" (final users): that makes asymmetry a key factor of data exchange.

In [21] both of these shortcomings are addressed by transferring a traditional SIS model onto a directed random graph, and the important effects of the topology of the graph on propagation speed are analyzed. The authors describe the behavior of virus infections on *sparse* and *local* graphs. In a sparse graph, each node has a small, constant average degree; on the contrary, in a local graph, the probability of having a vertex between nodes B and C is significantly higher if both have a vertex connected to the same node A. The authors discuss that in the landscape of the beginnings of the 90s the latter situation approximated very well the interaction between computer users. Among other results, it is shown that the more sparse a graph is, the slower is the spread of an infection on it; and the higher is the probability that an epidemic condition does not occur at all, which means that sparseness helps in containing *global* virus spread (while local spread is unhindered). Further elaborations on this type of model can be found in [23].

These findings are useful and interesting. However, it must be noted that often a SIR model, in which a "cured" system is not susceptible any more, could approximate better the behavior of many real cases of propagation when a patch or antivirus signature is available. Also, the introduction of the Internet as a convenient and immediate way for software and data exchange has arguably made the assumptions of locality and sparseness of the graph no longer valid.

3.3 Open questions in modeling mass-mailers

With the widespread adoption of the Internet, mass-mailing worms began to appear. The damage caused by *Melissa* virus in 1999, *Love Letter* in 2000 and *Sircam* in 2001 demonstrated that tricking users into executing the worm code attached to an e-mail, or exploiting a vulnerability in a common e-mail client to automatically launch it, is a successful way to propagate viral code.

In a technical report [24] Zou et al. describe a model of e-mail worm propagation. The authors model the Internet e-mail service as an undirected graph of relationships between people. In order to build a simulation of this graph, they assume that each node degree is distributed on a power-law probability function, an assumption drawn by the analysis of distribution of discussion group sizes, which result to be heavy-tailed: since adding a group to the address book adds an edge towards all components of the group, the distribution of node degree results heavy tailed too. Nowadays, discussion groups proactively filter attachments, so this assumption is challenged. Additionally, the authors employ a "small world" network topology, which seems to ignore completely the existence of interest groups and organizations, which naturally create clusters of densely connected vertexes. All these simplifications should be addressed in creating a good model of mass mailer propagation.

Furthermore, the authors assume that each user "opens" an incoming virus attachment with a fixed probability, different for each user but constant in time. This does not describe very well the typical behavior of users. Indeed, most experienced users avoid virus attachments altogether, while unexperienced users open them easily, at least the first time.

Additionally, it is observed that since user e-mail checking time is much larger than the average e-mail transmission time, the latter can be disregarded in the model. Since the overall spread rate of viruses gets higher as the variability of users' e-mail checking times increases, reliable statistics describing this process should be used in order to build better models of mass-mailer propagation.

Finally, when trying to determine the volume of messages generated by a mass mailer the fact that, in most cases, e-mail viruses install themselves as startup services on the system, and spread themselves at each opportunity, should be taken into account and properly modeled.

3.4 Open questions in modeling scanning worms

The concept of a self-contained, self-propagating program which does not require an host program to be carried around, was also developed early, but was somehow neglected for a long time. In 1988, however, the *Internet Worm* [25] changed the landscape of the threats. The Internet Worm was the first successful example of a self-propagating program which did not infect host files, but was self contained. Moreover, it was the first really successful example of an active network worm, which propagated on the Internet by using well-known vulnerabilities of the UNIX operating system. Other worms used open network shares, or exploited vulnerabilities in operating systems and server software to propagate.

The Random Constant Spread (RCS) model [18] was developed by Staniford, Paxson and Weaver using empirical data derived from the outbreak of the *Code Red* worm, a typical random scanning worm which propagates by using the *.ida vulnerability* discovered by eEye itself on June 18th 2001 [26], thus infecting vulnerable web servers running Microsoft IIS version 4.0 and 5.0. When Code Red infects an host, it spreads by launching 99 threads, which randomly generate IP addresses (excluding subnets 127.0.0.0/8, loopback, and 224.0.0.0/8, multicast) and try to compromise the hosts at those addresses using the same vulnerability.

A particularity of this worm is that it does not reside on the file system of the target machine, but it is carried over the network as the shellcode of the buffer overflow attack [27] it uses. When it infects an host, it resides only in memory: thus a simple reboot eliminates the worm, but does not avoid reinfection. Applying a patch to fix the IIS server or using temporary workarounds (e.g. activating a firewall, or shutting down the web server) makes instead the machine completely invulnerable to the infection. Thus, in order to model completely the worm we would need a SIR model where from I state we can either go to S or R state.

However, the RCS model makes a big approximation: it ignores that systems can be patched, powered and shut down, deployed or disconnected. In other words it is a simple SI model, with no recovery or immunization processes. This is only partially reasonable and justified by the speed of the worm propagation: in other words, the authors implicitly assume that the worm will peak before a remedy begins to be deployed.

An additional, more crucial approximation, is that the Internet topology is considered an undirected complete graph. In truth, the Internet being (as S. Breidbart defined it) "the largest equivalence class in the reflexive, transitive, symmetric closure of the relationship *can be reached by an IP packet from*", it is all but completely connected. In fact, re-

cent researches [28] show that as much as the 5% of the routed (and used) address space is not reachable by various portions of the network, due to misconfiguration, aggressive filtering, or even commercial disputes between carriers.

Let N be the total number of vulnerable servers which can be potentially compromised *from* the Internet. Let K be the average compromise rate, i.e. the number of vulnerable hosts that an infected host can compromise on average per unit of time at the beginning of the outbreak. K averages out any difference in processor speed, network bandwidth and location of the infected host. The model also assumes that a machine cannot be compromised multiple times and that, being 2^{32} a very large address space, the chance that two different instances of the worm *simultaneously* try to infect a single target is negligible. If $a(t)$ is the proportion of vulnerable machines which have been compromised at the instant t, the RCS model is described by the simple differential equation:

$$\frac{da}{dt} = Ka(1-a) \tag{4}$$

The solution of this equation is the well-known *logistic curve*. In [18] the authors fit their model to the "scan rate", or the total number of scans seen at a single site, instead than using the number of distinct attacker IP addresses, because this latter variable is distorted by time skew, unless the outbreak is observed from a very large address space, a concept known as a "network telescope" [29]. Researchers from CAIDA used data from such a telescope to describe the Code Red outbreak [30]. A total of about 359.000 hosts were infected by CRv2 in about 14 hours of activity. The worm was peaking when the self-deactivation mechanism it contained shut it down.

However, when we deal with UDP-based worms such as *Slammer* (which propagates by exploiting a buffer overflow vulnerability in Microsoft SQL Server) a radical change happens. Slammer had a doubling time of $8.5(\pm 1)$ seconds, while Code Red had a doubling time of about 37 minutes. Slammer infected more than 90 percent of vulnerable hosts within the first 10 minutes. This is caused by the fact that TCP based worms have to establish a connection before actually exploiting the vulnerability: having to wait for answers, they are *latency limited*. UDP based worm, on the contrary, scan at the full speed allowed by the network bandwidth available, so they are *bandwidth limited*.

Slammer's spreading strategy is based on random scanning, similarly to Code Red. Thus, the RCS model should fit its growth, but it fails after

a while. A common explanation for this failure is that the model does not take into due account bandwidth limitations on the global network: in other words, the failure and overload of links during worm propagation make the "global reachability" assumption less and less realistic as time goes on.

In [31] the RCS model was extended, creating a compartment-based model, in order to take into account the existence of bottleneck Internet links. The propagation equation becomes thus a system of nonlinear differential equations:

$$\left\{ \frac{da_i}{dt} = \left[a_i K \frac{N_i}{N} + Q_i \sum_{j \neq i} Q_j \frac{N_j}{N} a_j K \right] (1 - a_i) \right. \tag{5}$$

where we denote with N_i the number of susceptible hosts in the i-th compartment (AS_i), with a_i the proportion of infected hosts in the same compartment. We also suppose, for simplicity, that the average propagation speed K is constant in each compartment. Q_i, $0 < Q_i \leq 1$ is the fraction of attack packets that actually can get through the link of the i-th compartment, and is a rough approximation of the bottleneck effect of the Internet links. Numerical simulations of the equation and its effects on the global growth of the worm and on the observation of the growth from a telescope are also presented.

This model is derived for a set of compartments with a single connection to the rest of the world, which is only partially realistic. A model for multi-homed compartments that are not just leaves, but that forward traffic following realistic Internet policies would be desirable.

Zou et al. [32] propose a different approach for modeling slow worms such as Code Red incorporating the Kermack-Mckendrick model for host disinfection into the RCS equations. Additionally, the authors propose that the infection rate K should be considered a function of time, because of intervening network saturation and router collapse. Basically they rewrite the model as:

$$\frac{da}{dt} = K(t)\, a\, (1 - a - q - r) - \frac{dr}{dt} \tag{6}$$

where $q(t)$ is the proportion of susceptible hosts that are immunized at time t, and $r(t)$ is the proportion of infected hosts that are cured and immunized at time t. This model is thus called the *two-factor worm model*. In order to complete the model, the authors make some debatable assumptions on $q(t)$ and $r(t)$. In particular (similarly to the kill signal theory

described in [33]), the patching process is modeled as a "counter-worm":

$$\frac{dq}{dt} = \mu(1 - a - q - r)(a + r)$$

This equation is somehow arbitrary, and further analysis on the two-factor model is needed before it can be considered a sound model of viral propagation.

3.5 Other open questions in propagation modeling

Some authors [34] have explored discrete time models, in the hope to better capture the discrete time behavior of a worm. However, a continuous model is appropriate for such large scale models, and the epidemiological literature is clear in this direction. The benefits of using a discrete time model seem very limited, but this is difficult to say since the base assumptions of this particular model are not completely correct. More exploration of the usage of discrete time models could lead to interesting results.

It is important to note that modern viruses often use a mix of different techniques to spread (for instance, Sircam uses both mass mailing and open network shares, while *Nimda* uses four different mechanisms to propagate). We are not aware, however, of any existing model which takes into account multi-vector viruses and worms.

4 Open problems in antiviral countermeasures

4.1 Monitoring and early warning

In current infrastructure where worms are able to achieve quick penetration it is essential to research and develop methods for prevention that will prevent attacks as early as possible. For example, Ibrahim and al. [35] demonstrate this approach by proactive email worm prevention.

Because of the effects of distortion described in Section 3.4, in [36] the models of active worm propagation are used to build an early monitoring and alerting system for TCP or UDP based worms, based on distributed *ingress* and *egress* sensors for worm activity. A data collection engine based on a Kalman filter is used to create an alerting system, capable of reliably setting off alarms as early as when the proportion of infected system is $1\% \leq a \leq 2\%$. It is also shown that this early warning method works well also with fast spreading worms, and even if an hit-list startup strategy is used.

However, we need more research in areas of proactive prevention. In general interesting areas could be network forensics, detecting infected host systems and preventing malicious operations from infected hosts.

4.2 Virus resistant infrastructures

If we develop further the concept of proactive prevention we may end up in research that will promote prevention as an inherent part of infrastructure design. We may find examples of such attempts from the development of IPv6 (Internet Protocol version 6), processor architecture design and buffer overflow [37] prevention techniques. However, we still need more holistic approaches. For example, security can be an inherent part of computer architecture and network architecture design [38]. Interesting questions may arise from construction of virus resistant and self-defending architectures.

4.3 Integrity verification

Viruses are a violation against system integrity. Unfortunately, in current systems integrity is difficult to verify and operating environments seldom support systematic integrity verification. There are solutions for system integrity verification, but integrity verification is not typically adapted as an inherent part of system design.

Radai established theory of integrity verification related to computer virology [39, 40]. Furthermore, Bontchev presented some methods viruses can use to attack integrity checking programs and how the attacks could be prevented [41]. More recently, Filiol [9, Chap. 8] technically demonstrated how integrity checking can be bypassed. One interesting question could be: how to adapt integrity verification as securely as possible against malware attacks? For example, new information system architectures may be needed to support integrity verification.

4.4 Effects of Quarantine

Quarantine is the world's oldest defense against viruses. In [42] a dynamic preventive quarantine system is proposed, which places suspiciously behaving hosts under quarantine for a fixed interval of time. Models and simulation of a quarantine system are proposed, however such a system would be difficult to deploy. Since hosts cannot be trusted to auto-quarantine themselves, on most networks quarantine would act on remotely manageable enforcement points (i.e. firewalls and intelligent network switches).

Since these components are limited, entire blocks of network would need to be isolated at once, increasing the probability that innocent hosts will be denied service as a side effect of the quarantine system.

In addition, as shown in [31], virus spread is not stopped but only slowed down inside each quarantined block. Moreover, it should be considered that the "kill signal" effect (i.e. the distribution of anti-virus signatures and patches) would be hampered by aggressive quarantine policies (something which is not taken into account in the modified Kerman-McKendrick models presented in [42]).

In [43] various containment strategies (content filtering and blacklisting) are simulated, deriving lower and upper bounds of efficacy. Albeit interesting, the results on blacklisting share the same weakness pointed out before: it's not realistic to think about a global blacklisting engine, enforced at network level.

More research on practical quarantining systems are needed in order to bring these approaches into real-world use. On a LAN, an intelligent network switch could be used to selectively shut down the ports of infected hosts, or to cut off an entire sensitive segment. Network firewalls and perimeter routers can be used to shut down the affected services. Reactive IDSs (the so-called "intrusion prevention systems") can be used to selectively kill worm connections based on attack signatures. Automatic reaction policies, however, are intrinsically dangerous. False positives and the possibility of fooling a prevention system into activating a denial-of-service are dangerous enough to make most network administrators wary.

4.5 Immunization

In [33] the effect of selective immunization of computers on a network is discussed. The dynamics of infection and the choice of immunization targets are examined for two network topologies: a hierarchical, tree-like topology (which is obviously not realistic for modeling the Internet), and a cluster topology. The results are interesting, but the exact meaning of "node immunization" is not defined. While such a study could be used to prioritize the process of patching on a widespread network, unless some new ideas for virus prevention are proposed, the practical possibilities of application for such a model seem extremely limited.

4.6 Honeypots and tarpits

Honeypots are fake computer system and networks, used as a decoy to cheat intruders. They are installed on dedicated machines, and left as a

bait so that aggressors will lose time attacking them and trigger an alert. Since honeypots are not used for any production purpose, any request directed to the honeypot is at least suspect. Honeypots can be made up of real sacrificial systems, or of simulated hosts and services (created using Honeyd by Niels Provos, for example).

A honeypot could be used to detect the aggressive pattern of a worm through anomaly detection: since honeypots are empty of true users, any non-simulated traffic hitting them is suspicious. Repeated connections towards the same ports of the honeypot machines are a good indicator of a scanning worm at work. The honeypot can thus be used as an alerting system. Also, once a worm has entered a honeypot, its payload and replication behaviors can be easily studied, provided that an honeywall is used to quarantine the sacrificial hosts making them unable to actually attack the real hosts outside.

As an additional possibility, an honeypot can be used to slow down worm propagation, particularly in the case of TCP-based worms. By delaying the answers to the worm connections, a honeypot may be able to slow down its propagation; very much the same technique used in the LaBrea "tarpit" tool, which replies to any connection incoming on an unused IP address of a network, and simulates a TCP session with the possible aggressor. LaBrea slows down the connection: when data transfer begins, the TCP window size is set to zero, so that no data can be transferred. The connection is kept open, and any request to close the connection is ignored. This means that the worm will have to wait for a timeout in order to disconnect, since it uses the standard TCP stack of the host machine which follows RFC standards. A worm won't be able to detect this slowdown, and if enough fake targets are present, its growth will be slowed down. Obviously, a multi-threaded worm will be less affected by this technique. This effect should be properly studied and modeled to evaluate its effectiveness.

4.7 Counterattacks and good worms

Counterattack may seem a viable cure to worms. When host A sees an incoming worm attack from host B, it knows that host B must be vulnerable to the particular exploit that the worm uses to propagate (unless the worm itself removed that vulnerability as a result of infection). By using the same type of exploit, host A can automatically take control of host B and try to cure it from infection and patch it.

The first important thing to note is that, fascinating as the concept may seem, this is not legal, unless host B is under the control of the same

administrator of host A. Additionally, automatically patching a remote host is always a dangerous thing, which can cause considerable unintended damage (e.g. breaking services and applications that rely on the patched component).

Another solution which in past proved to be worse than the illness is the release of a so-called "good" or "healing" worm, which automatically propagates in the same way the bad worm does, but carries a payload which patches the vulnerability. A good example of just how dangerous such things may be is the *Welchia* worm, which was meant to be a cure for *Blaster*, but actually caused devastating harm to the networks. Such proposals must be carefully evaluated, as was done in [44]

5 Technical and practical research areas in computer virology

Antivirus software evaluation It is very difficult to accurately evaluate the quality and the limitations inherent to the different antiviral products available today. Users can only compare marketing claims of each vendor, without any real information about the detection and disinfection power and efficiency. Notoriety, or market share, could be taken as an indicator. The raw percentage of viruses/malware that are effectively detected and efficiently disinfected can also be considered. But a little experience in antiviral software quickly shows that this approach is quite sterile.

The problem of having precise and efficient technical evaluation tools, and a clear methodology to use them, is of the highest importance. This problem must be considered in connection with a crucial property expressing the complexity virus writers must face to obtain technical information about the antivirus during a "black box analysis" process. The analysis of viral databases is probably the best example.

Malware taxonomy and phylogeny As we have seen multiple times, recursive self-replication is the fundamental characteristic of a virus. However, when we go beyond that it becomes difficult to classify malware. Even a definition for the term "computer worm" has not been agreed on. For example, if we define that a virus must infect a host and a worm is self-contained, the meaning of "host" must be discussed. When we reach terms like "Trojan horse" and "spyware" the precise definitions is even more difficult. At least, the following reasons can be found:

1. Malicious intentions are difficult (and sometimes impossible) to predict by analyzing program code.
2. A program may be used maliciously even when it is designed for beneficial purposes, and vice-versa.
3. There exist a number of "gray areas" where it is impossible to say whether a program belongs to a certain category or not.

Despite the difficulties in defining malware, research on objective definitions and criteria for classification is needed. Brunnstein proposes an interesting classification scheme based on software disfunctions [45]. However, we also need practical definitions. In general interesting questions could be: what are different malware categories and sub-categories? What are the functionalities for a certain program category? What are the precise definitions? How to prove that a program code belongs to a certain category?

Even the naming convention of malware is still an open problem, perhaps one of the most crucial problems in modern computer virology. Unfortunately nobody proved that this is not an "undecidable problem". It is a matter of fact, however, that every antivirus company develops its own naming convention, ignoring the other ones. Very frequently, all these naming conventions appear to be at least partially incompatible, but unless a sound and rational classification base is developed, nobody will accept to give up. Recent developments [46, 47] have shown that phylogeny models – *i.e.* taking into account the fact that programs may be evolved through code rearrangements or that viruses are rarely written from scratch and are mostly derived from known previous codes – is likely to produce the desired tools for a unified naming convention. But many problems still exist. The authors of [47] focused on permutations of code. They have identified some questions that are still to be solved. Moreover, they only consider sequence-based phylogeny models. Would it be possible to extend their approach to function-based phylogeny?

5.1 Malware in smart phones

Even a modest cellular phone includes software that controls the phones operations. Meanwhile phones are getting more and more properties of computers: connectivity, applications and calculation power. Although in Symbian smart phone operating system security is part of the design vulnerabilities may still remain. Niemelä presents technical aspects of Symbian from the malware point of view [48] and Reynaud-Plantey [49] recently analyzed some new aspects of the viral risk with respect to

the Java language. MMS (Multimedia Messaging System), Bluetooth and vulnerabilities enable existence of viruses.

Research in computer virology is so far occasional in the area of smart phones. Still smart phones bring special aspects to research: mobility, cost of services and fixed wireless connections.

6 Conclusions

We have proposed some of the most interesting open research problems and areas in computer virology, with an emphasis on theoretical aspects. To begin, we focused on theoretical computer virology, presenting the core results already developed in literature, and the problems that are still waiting a solution. In particular, complexity problems, virus classification and new classes of viruses still need much research.

Virus propagation modeling techniques als need improvement in order to capture new trends in the propagation of common viruses, mass-mailers and random scanning worms. Proposed countermeasures are also described, along with open questions: how can they be validated before being implemented? Which new defensive techniques do we need against the next generations of aggressive malware?

Finally, we presented practical and technical research areas, to complete our review of open research issues: we focused on those problems that, in our view, could benefit from a more theoretically sound approach.

Of course, we have not addressed all open problems. For instance, there are interesting issues concerning programming languages, their semantics and computer viruses. We could wonder whether it is possible to develop a high-level programming language compiler which guarantees that no attacks can be performed. This type of questions is generally addressed in computer safety research, but will likely be deeply interesting in defeating computer malware.

In conclusion, since the research domain in computer virology is a new one, we can expect fundamental research outcomes to be found in the next few years, and to deeply influence the future of computer security technologies for virus defense.

Acknowledgments

We would like to thank Jean-Yves Marion for his valuable comments and his help in improving this paper. He very kindly helped in developing some of the points of this paper, and in particular pointed out the research trend on computer language safety.

References

1. Fred Cohen. *Computer Viruses*. PhD thesis, University of Southern California, 1985.
2. Leonard M. Adleman. An abstract theory of computer viruses. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO'88 - Lecture Notes in Computer Science 403*, pages 354–374. Springer-Verlag, 1988.
3. Zhihong Zuo and Mingtian Zhou. Some further theoretical results about computer viruses. *The Computer Journal*, 47(6), 2004.
4. Guillaume Bonfante, Mathieu Kaczmarek, and Jean-Yves Marion. On abstract computer virology from a recursion-theoretic perspective. *Journal in Computer Virology*, 1(3-4), 2005.
5. Guillaume Bonfante, Mathieu Kaczmarek, and Jean-Yves Marion. Toward an abstract computer virology. In *Proceedings of the ICTAC'05, Lecture Notes in Computer Science 3722*, pages 579–593. Springer Verlag, 2005.
6. Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, 1967.
7. Diomedis Spinellis. Reliable identification of bounded-length viruses is np-complete. *IEEE Transactions in Information Theory*, 49(1), 2003.
8. Christos H. Papadimitriou. *Complexity Theory*. Addison Wesley, 1994.
9. Eric Filiol. *Computer Viruses: from Theory to Applications*. Springer-Verlag, 1 edition, 2005.
10. Zhihong Zuo and Mingtian Zhou. On the time complexity of computer viruses. *IEEE Transactions in Information Theory*, 51(8), 2003.
11. David M. Chess and Steve R. White. An undetectable computer virus. In *Proc. Virus Bulletin Conference*, 2000.
12. Eric Filiol. *Advanced Viral Techniques: Mathematical and Algorithmic Aspects*. Springer-Verlag, To appear, 2006.
13. Neil D. Jones. *Computability and complexity: from a programming perspective*. MIT Press, 1997.
14. F. Jessie MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
15. Mika Hirvensalo. *Quantum Computing*. Springer-Verlag, 2nd edition, 2004.
16. Gilles Brassard. A bibliography of quantum cryptography. *SIGACT News*, 24(3):16–20, 1993.
17. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. of the 35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1994.
18. Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium (Security '02)*, 2002.
19. Ian Whalley, Bill Arnold, David Chess, John Morar, Alla Segal, and Morton Swimmer. An environment for controlled worm replication and analysis. In *Proceedings of the Virus Bulletin Conference*, September 2000.
20. Steve R. White. Open problems in computer virus research. In *Proceedings of the Virus Bulletin Conference*, Oct 1998.
21. Jeff O. Kephart and Steve R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Symposium on Security and Privacy*, pages 343–361, 1991.

22. Herbert W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, 2000.
23. Lora Billings, William M. Spears, and Ira B. Schwartz. A unified prediction of computer virus spread in connected networks. *Physics Letters A*, (297):261–266, 2002.
24. Cliff Changchun Zou, Don Towsley, and Weibo Gong. Email virus propagation modeling and analysis. Technical Report TR-CSE-03-04, University of Massachussets, Amherst.
25. Eugene H. Spafford. Crisis and aftermath. *Communications of the ACM*, 32(6):678–687, 1989.
26. Ryan Permeh and Riley Hassell. Microsoft I.I.S. remote buffer overflow. Advisory AD20010618, June 2001.
27. Elias 'Aleph1' Levy. Smashing the stack for fun and profit. *Phrack magazine*, 7(49), November 1996.
28. Abha Ahuja Craig Labovitz and Michael Bailey. Shining light on dark address space. Technical report, Arbor networks, Nov 2001.
29. David Moore. Network telescopes: Observing small or distant security events. In *Proceedings of the 11th USENIX Security Symposium*, Aug 2002.
30. David Moore, Colleen Shannon, and Jeffery Brown. Code-red: a case study on the spread and victims of an internet worm. In *Proceedings of the ACM SIGCOMM/USENIX Internet Measurement Workshop*, Nov 2002.
31. Giuseppe Serazzi and Stefano Zanero. Computer virus propagation models. In Maria Carla Calzarossa and Erol Gelenbe, editors, *Tutorials of the 11th IEEE/ACM Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecom. Systems - MASCOTS 2003*. Springer-Verlag, 2003.
32. Cliff Changchun Zou, Weibo Gong, and Don Towsley. Code red worm propagation modeling and analysis. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 138–147. ACM Press, 2002.
33. Chenxi Wang, John C. Knight, and Matthew C. Elder. On computer viral infection and the effect of immunization. In *ACSAC '00: Proceedings of the 16th Annual Computer Security Applications Conference*, page 246, Washington, DC, USA, 2000. IEEE Computer Society.
34. Zesheng Chen, Lixin Gao, and Kevin Kwiat. Modeling the spread of active worms. In *Proceedings of IEEE INFOCOM 2003*, 2003.
35. Ibrahim K. El-FarArun, Richard Ford, Attila Ondi, and Manan Pancholi. Suppressing the spread of email malcode using short-term message recall. *Journal in Computer Virology*, 1(3–4), 2005.
36. Cliff Changchun Zou, Lixin Gao, Weibo Gong, and Don Towsley. Monitoring and early warning for internet worms. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 190–199. ACM Press, 2003.
37. Chien Eric and Szr Pter. Blended attacks: Exploits, vulnerabilities and buffer-overflow techniques in computer viruses. In *Proceedings of Virus Bulletin Conference 2002*, pages 1–35. Virus Bulletin Ltd., 2002.
38. Helenius Marko. Realisation ideas for secure system design. In U.E. Gattiker, editor, *EICAR Conference Best Paper Proceedings*, Copenhagen, 2003. EICAR.
39. Radai Yisrael. Checksumming techniques for anti-viral purposes. In *Proceedings of First International Virus Bulletin Conference*, 1991.
40. Radai Yisrael. Integrity checking for anti-viral purposes: Theory and practice. improved version of earlier conference paper, 1994.

41. Bontchev Vesseli. Possible virus attacks against integrity programs and how to prevent them. In *Proceedings of 2nd International Virus Bulletin Conference*, pages 131–141, 1992.
42. Cliff Changchun Zou, Weibo Gong, and Don Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *Proceedings of the ACM CCS Workshop on Rapid Malcode (WORM'03)*, Oct 2003.
43. David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of IEEE INFOCOM*, 2003.
44. Frank Castaneda, Emre Can Sezer, and Jun Xu. Worm vs. worm: preliminary study of an active counter-attack mechanism. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 83–93, New York, NY, USA, 2004. ACM Press.
45. Klaus Brunnstein. From antivirus to antimalware software and beyond: Another approach to the protection of customers from dysfunctional system behaviour. In *Proceedings of 22nd National Information Systems Security Conference*, 1999.
46. Leslie Ann Goldberg, Paul W. Goldberg, Cynthia A. Phillips, and Gregory B. Sorkin. Constructing computer virus phylogenies. *Journal of Algorithms*, 26:188–208, 1998.
47. Md. Enamul Karim, Andrew Walenstein, and Arun Lakhotia. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1-2), 2005.
48. Niemelä Jarmo. What makes symbian malware tick. In *Proceedings of Virus Bulletin Conference*, pages 115–120, England, 2005. Virus Bulletin Ltd.
49. Daniel Reynaud-Plantey. New threats of java viruses. *Journal in Computer Virology*, 1(3–4), 2005.